2019

Random Sampling for Group-By Queries

Trong Duc Nguyen
Iowa State University, trong@iastate.edu

Ming-Hung Shih
Iowa State University, mshih@iastate.edu

Sai Sree Parvathaneni
Iowa State University, ssree@iastate.edu

Bojian Xu
Eastern Washington University

Divesh Srivastava
AT&T Labs–Research

See next page for additional authors

Random Sampling for Group-By Queries

Abstract
Random sampling has been widely used in approximate query processing on large databases, due to its potential to significantly reduce resource usage and response times, at the cost of a small approximation error. We consider random sampling for answering the ubiquitous class of group-by queries, which first group data according to one or more attributes, and then aggregate within each group after filtering through a predicate. The challenge with group-by queries is that a sampling method cannot focus on optimizing the quality of a single answer (e.g. the mean of selected data), but must simultaneously optimize the quality of a set of answers (one per group).
We present CVOPT, a query- and data-driven sampling framework for a set of group-by queries. To evaluate the quality of a sample, CVOPT defines a metric based on the norm (e.g. $\ell 2$ or $\ell \infty$) of the coefficients of variation (CVs) of different answers, and constructs a stratified sample that provably optimizes the metric. CVOPT can handle group-by queries on data where groups have vastly different statistical characteristics, such as frequencies, means, or variances. CVOPT jointly optimizes for multiple aggregations and multiple group-by clauses, and provides a way to prioritize specific groups or aggregates. It can be tuned to cases when partial information about a query workload is known, such as a data warehouse where queries are scheduled to run periodically.
Our experimental results show that CVOPT outperforms current state-of-the-art on sample quality and estimation accuracy for group-by queries. On a set of queries on two real-world data sets, CVOPT yields relative errors that are 5x smaller than competing approaches, under the same space budget.

Disciplines
Computer Sciences | Electrical and Computer Engineering

Comments
This is a pre-print of the article Nguyen, Trong Duc, Ming-Hung Shih, Sai Sree Parvathaneni, Bojian Xu, Divesh Srivastava, and Srikanta Tirthapura. "Random Sampling for Group-By Queries." *arXiv preprint arXiv:1909.02629* (2019). Posted with permission.

Authors
Trong Duc Nguyen, Ming-Hung Shih, Sai Sree Parvathaneni, Bojian Xu, Divesh Srivastava, and Srikanta Tirthapura

Random Sampling for Group-By Queries

Trong Duc Nguyen[1] Ming-Hung Shih[1] Sai Sree Parvathaneni[1] Bojian Xu[2]
Divesh Srivastava[3] Srikanta Tirthapura[1]

[1]Iowa State University, IA 50011, USA
[2] Eastern Washington University, WA 99004, USA.
[3]AT&T Labs–Research, NJ 07921, USA

trong@iastate.edu mshih@iastate.edu ssree@iastate.edu bojianxu@ewu.edu
divesh@research.att.com snt@iastate.edu

September 13, 2019

Abstract

Random sampling has been widely used in approximate query processing on large databases, due to its potential to significantly reduce resource usage and response times, at the cost of a small approximation error. We consider random sampling for answering the ubiquitous class of group-by queries, which first group data according to one or more attributes, and then aggregate within each group after filtering through a predicate. The challenge with group-by queries is that a sampling method cannot focus on optimizing the quality of a single answer (e.g. the mean of selected data), but must simultaneously optimize the quality of a set of answers (one per group).

We present CVOPT, a query- and data-driven sampling framework for a set of queries that return multiple answers, e.g. group-by queries. To evaluate the quality of a sample, CVOPT defines a metric based on the norm (e.g. ℓ_2 or ℓ_∞) of the coefficients of variation (CVs) of different answers, and constructs a stratified sample that provably optimizes the metric. CVOPT can handle group-by queries on data where groups have vastly different statistical characteristics, such as frequencies, means, or variances. CVOPT jointly optimizes for multiple aggregations and multiple group-by clauses, and provides a way to prioritize specific groups or aggregates. It can be tuned to cases when partial information about a query workload is known, such as a data warehouse where queries are scheduled to run periodically.

Our experimental results show that CVOPT outperforms current state-of-the-art on sample quality and estimation accuracy for group-by queries. On a set of queries on two real-world data sets, CVOPT yields relative errors that are 5× smaller than competing approaches, under the same budget.

1 Introduction

As data size increases faster than the computational resources for query processing, answering queries on large data with a reasonable turnaround time has remained a significant challenge. One solution to handling this data deluge is through random sampling. A sample is a subset of data, collected with the use of randomness to determine which items are included in the sample and which

are not. A query posed on the data can be quickly and approximately answered by executing the query on the sample, followed by an appropriate normalization. Sampling is attractive when a good trade-off can be obtained between the size of the sample and the accuracy of the answer.

We investigate random sampling to support "group-by" queries. A group-by query partitions an input relation into multiple groups according to the values of one or more attributes, and applies an aggregate function within each group. For instance, on a relation consisting of student records `Student(name, id, year, major, gpa)`, the following SQL query asks for the average `gpa` of each major:

```
SELECT   major, AVG(gpa)
FROM     Student
GROUP BY major
```

Group-by queries are very common, especially in data warehouses. For instance, in the TPC-DS benchmark [15], 81/ 99 queries have group-by clause. We address the basic question: how to sample a table to accurately answer group-by queries?

A simple method is *uniform sampling*, where each element is sampled with the same probability. It has been recognized that uniform sampling has significant shortcomings [4]. Since a group will be represented in proportion to its volume (i.e. the number of elements in the group), a group whose volume is small may have very few elements selected into the sample or may be missing altogether, while a group whose volume is large may have a large number of elements in the sample. This can clearly lead to high errors for some (perhaps a large fraction of) groups, as confirmed by our experimental study.

The accuracy of group-by queries can be improved using stratified sampling [1, 5, 20]. Data is partitioned into multiple strata. Uniform sampling is applied within each stratum, but different probabilities may be used across strata. The "best" way to use stratified sampling for group-by queries is not obvious – how to stratify the data into groups, and how to assign a sampling probability to a stratum?

Prior work on "congressional sampling" [1] has advocated the use of a fixed allocation of samples to each stratum, irrespective of its size (the "senate" strategy)[1]. Consider the above example, where the table is grouped by the attribute `major`. Suppose the population was stratified so that there is one stratum for each possible value of `major`, and uniform sampling is applied within each stratum. Even for this simple example, congressional sampling can lead to a sub-optimal allocation of samples. Consider two groups, 1 and 2 with the same number of elements and the same mean, but group 1 has a higher variance than group 2. In estimating the mean of each stratum, allocating equal numbers of samples to each group leads to a worse error for group 1 than for group 2. It is intuitively better to give more samples to group 1 than to group 2, but it is not clear how many more. *Thus far, there has been no systematic study on sample allocation to groups, and our work aims to fill this gap.*

Our high-level approach is as follows. We first define a metric that quantifies the error of an estimate for a group-by query through using the sample, and we obtain an allocation and then a sample that minimizes this metric. Such an optimization approach to sampling has been used before. For instance, the popular "Neyman allocation" [16, 17] assigns samples to strata to *minimize the variance of an estimate of the population mean* that can be derived from the sample. The challenge with group-by queries is that there isn't a single estimate to focus on, such as the population mean, but instead, an answer for each group.

[1]More accurately, a hybrid of fixed and population-based allocations.

Since multiple estimates are desired, we combine the metrics corresponding to different estimates to derive a global metric for the quality of a sample. One possible global metric is the sum of the variances of the different estimators. Such a global metric is inadequate in the case when the means of different groups are vastly different, in which case variances cannot be compared across groups. For instance, consider 2 groups with means $\mu_1 = 1000$ and $\mu_2 = 100$. Suppose an estimator for each group has a variance of 100. Though both estimators have the same variance, they are of different quality. If data within each group follows a normal distribution, then we can derive that with a probability greater than 96%, the estimated mean y_1 is between $[950; 1050]$ and the estimated mean y_2 is between $[50; 150]$. In other words, $y_1 = \mu_1 \pm 5\%$ and $y_2 = \mu_2 \pm 50\%$, and therefore y_1 is a much better estimate than y_2, in terms of relative accuracy. However, since the global metric adds up the variances, both the estimators contribute equally to the optimization metric. This leads to an allocation that favors groups that have a large mean over groups that have a small mean.

Coefficient of Variation (CV): In order to combine the accuracies of different estimators, perhaps with vastly different expectations, it is better to use the *coefficient of variation* (also called "relative variance"). The coefficient of variation of a random variable X is defined as $\mathbb{CV}[X] = \frac{\mathbb{S}[X]}{\mathbb{E}[X]}$, where $\mathbb{E}[X]$ is the expectation (mean) and $\mathbb{S}[X]$ the standard deviation of X, respectively. We assume that the attribute that is being aggregated has a non-zero mean, so the CV is well defined. The CV of a random variable X is directly connected to its relative error $r(X) = |X - \mathbb{E}[X]|/\mathbb{E}[X]$ as follows. For a given $\epsilon > 0$, using Chebyshev's inequality, we have $\Pr[r(X) > \epsilon] = \Pr[|X - \mathbb{E}[X]| > \epsilon\mathbb{E}[X]] \leq \frac{\mathrm{Var}X}{\epsilon^2\mathbb{E}[X]^2} = \left(\frac{\mathbb{CV}[X]}{\epsilon}\right)^2$. Smaller the CV, the smaller is the bound for the relative error of a given estimator. Our approach is to choose allocations to different strata so as to optimize a metric based on the CVs of different per-group estimators.

1.1 Contributions

(1) We present `CVOPT`, a novel framework for sampling from data which is applicable to any set of queries that result in multiple answers whose errors need to be simultaneously minimized (a specific example is a group-by query). `CVOPT` optimizes a cost function based on the weighted aggregate of the CVs of the estimates that are desired by the query. We present an algorithm (also called `CVOPT`) that computes a *provably optimal* allocation. To our knowledge, this is the first algorithm that results in a provably optimal sample for group-by queries. We consider two ways of aggregating CVs – one based on the ℓ_2 norm of the CVs, and another based on the ℓ_∞ norm (the maximum) of the CVs.

(2) `CVOPT` can be adapted to the following circumstances, in increasing order of generality:

• With a single aggregation on a single method of grouping attributes (SASG), `CVOPT` leads to good statistical properties over a range of workloads. The *distribution of errors of different groups is concentrated around the mean error*, much more so than prior works [20, 1]. As a result, the expected errors of the different per-group estimates are approximately equal, while prior work can lead to some groups being approximated very well while other groups being approximated poorly.

• With multiple aggregations for the same group-by clause (MASG), and for the general case of multiple aggregations and multiple ways of group-by (MAMG), we provide a way to derive a sample that optimizes for a combined metric on all aggregate queries. A special case is the `Cube` query that is widely used in analytics and decision support workloads.

• The user is allowed to specify a weight to each answer, allowing her to prioritize different query/-group combinations, and use (perhaps uncertain) knowledge about the workload that may be

specified by a probability distribution.

(3) The samples can be reused to answer various queries that incorporate selection predicates that are provided at query time, as well as new combinations of groupings[2].

(4) We present a detailed experimental study on two real-world datasets (`OpenAQ` and `Bikes`) that show that `CVOPT` using the ℓ_2 norm of the CVs provides good quality estimates across all groups in a group-by query, and provides a relative error that is often up to $5\times$ smaller than prior work [1, 9, 20]. Further, no prior work can compete with `CVOPT` as a consistently second best solution for group-by queries.

1.2 Prior Work

The closest work to ours is that of Rösch and Lehner [20] (which we hence forth call `RL`), who propose non-uniform sampling to support group-by queries, where different groups can have vastly different variances. For the case of a single-aggregation and a single group-by, `CVOPT` is similar to `RL`, since `RL` also uses the coefficient of variation, in addition to other heuristics. The main difference is that `CVOPT` provably minimizes the ℓ_2 norm (or the ℓ_∞ norm) of the CVs, whereas `RL` is a heuristic without a guarantee on the error, and in fact does not have a well-defined optimization target. The provable guarantees of `CVOPT` hold even in case of multiple aggregations and/or multiple group-bys, whereas `RL` uses a variety of heuristics to handle sample allocation.

Another closely related prior work is "congressional sampling" [1] (which we call `CS`), which targets sampling for a collection of group-by queries, especially those that consider a "cube by" query on a set of group-by attributes. `CS` is based on a hybrid of frequency-based allocation (the "house") and fixed allocation (the "senate"). `CVOPT` differs from `CS` through using the coefficients of variation (and hence also the variances) of different strata in deciding allocations, while `CS` only uses the frequency. `CVOPT` results in a provably optimal allocation (even in the case of multiple group-by clauses and aggregates) while `CS` does not have such a provable guarantee.

A recent work "`Sample+Seek`" [9] uses a combination of measure-biased sampling and an index to help with low-selectivity predicates. Measure-biased sampling favors rows with larger values along the aggregation attribute. This does not consider the variability within a group in the sampling step – a group with many rows, each with the same large aggregation value, is still assigned a large number of samples. In contrast, `CVOPT` favors groups with larger CVs, and emphasizes groups with larger variation in the aggregation attribute. Further, unlike `CVOPT`, `Sample+Seek` does not provide a sampling strategy that is based on an optimization framework. Our work can potentially be used in conjunction with the index for low-selectivity queries. In our experimental section, we compare with `RL`, `CS`, and `Sample+Seek`.

All statistics that we use in computing the allocation, including the frequency, mean, and coefficient of variation of the groups, can be computed in a single pass through the data. As a result, the overhead of offline sample computation for `CVOPT` is similar to that of `CS` and `RL`. We present other related work from the general area of approximate query processing in Section 7.

Roadmap: We present preliminaries in Section 2, followed by algorithms for sampling for a single group-by query in Section 3 and algorithms for multiple group-by in Section 4. We present an algorithm for different error metrics in Section 5, a detailed experimental study in Section 6, followed by a survey of other related works, and the conclusions.

[2]Arbitrary new groupings and selection predicates are not supported with a provable guarantee. Indeed, we can show no sample that is substantially smaller than the data size can provide accurate answers for arbitrary queries, since one can reconstruct the original table by making such repeated queries.

2 Preliminaries

For a random variable X, let $\mathbb{E}[X]$ denote its expectation, $\mathbb{VAR}[X]$ its variance, $\mathbb{S}[X] = \sqrt{\mathbb{VAR}[X]}$ its standard deviation, and $\mathbb{CV}[X] = \frac{\mathbb{S}[X]}{\mathbb{E}[X]}$ its coefficient of variation.

The answer to a group-by query can be viewed as a vector of results, one for the aggregate on each group. In case multiple aggregates are desired for each group, the answer can be viewed as a two-dimensional matrix, with one dimension for the groups, and one for the aggregates. In this work, for simplicity of exposition, we focus on the aggregate `AVG`, i.e. the mean. Note that the sample can answer queries involving selection predicates provided at runtime (by simply applying the predicate on the sample) so that it is not possible to precompute the results of all possible queries. Aggregates such as median and variance can be handled using a similar optimization, and the same sample can be jointly optimized for multiple aggregate functions.

Let the groups of a given query be denoted $1, 2, \ldots, r$, and $\mu_1, \mu_2, \ldots, \mu_r$ denote the mean of value within each group. A sample of a table is a subset of rows of the table. From a sample, we derive estimators y_i, of μ_i, for each $i = 1 \ldots r$. We say that y_i is an unbiased estimator of μ_i if $\mathbb{E}[y_i] = \mu_i$. Note that an unbiased estimator does not necessarily mean an estimator that reliably estimates μ_i with a small error.

For a group-by query with r groups in data, numbered $1..r$, the aggregates can be viewed as an array $\mu = [\mu_1, \mu_2, \ldots, \mu_r]$, the estimates can be viewed as the vector $y = [y_1, y_2, \ldots, y_r]$, and coefficients of variation of the estimates as the vector $\mathbb{C} = [\mathbb{CV}[y_1], \mathbb{CV}[y_2], \ldots, \mathbb{CV}[y_r]]$. We first focus on the overall error equal to the ℓ_2 norm of the vector $\mathbb{C}$, defined as:

$$\text{error}(y) = \ell_2(\mathbb{C}) = \sqrt{\sum_{i=1}^{r} (\mathbb{CV}[y_i])^2}$$

Applying the above metric to the earlier example, we have $\mathbb{CV}[y_1] = 10/1000 = 0.01$, while $\mathbb{CV}[y_2] = 10/100 = 0.1$. This (correctly) evaluates y_2 as having a higher contribution to the overall error than y_1. If we were to optimize ℓ_2 norm of the vector $[\mathbb{CV}[y_1], \mathbb{CV}[y_2]]$, resources would be spent on making $\mathbb{CV}[y_2]$ smaller, at the cost of increasing $\mathbb{CV}[y_1]$. We argue this is the right thing to do, since all results in a group-by query are in some sense, equally important. If we know apriori that some results are more important than others, they can be handled by using a weighting function for results, as we describe further. We also consider the ℓ_∞ norm, defined as: $\ell_\infty(\mathbb{C}) = \max_{i=1}^{r} \mathbb{CV}[y_i]$.

We can also **assign weights** to different results in computing the error. Consider a set of positive real valued numbers, one for each result $i = 1 \ldots r$, $w = \{w_i\}$. The weighted ℓ_2 metric is: $\text{error}(y, w) = \ell_2(\mathbb{C}, w) = \sqrt{\sum_{i=1}^{r} w_i \cdot (\mathbb{CV}[y_i])^2}$, where a larger w_i indicates a higher accuracy demand for y_i.

3 Single Group-by

3.1 Single Aggregate, Single Group-By

The first case is when we have a single aggregate query, along with a single group-by clause. Note that grouping does not have to use a single attribute, but could use multiple attributes. For example, `SELECT year,major, AVG(gpa) FROM Student GROUP BY year,major`. *Given a budget*

of sampling M records from a table for a group-by query with r groups, how can one draw a random sample such that the accuracy is maximized?

We use stratified sampling. In the case of a single group-by clause, *stratification directly corresponds to the grouping.* There is a stratum for each group, identified by a distinct value of the combination of group-by attributes. In the above example, there is one stratum for each possible combination of the `(year, major)` tuple. Probabilities of selecting a record from the table are not necessarily equal across different strata, but are equal within a stratum.

One simple solution, which we call as `SENATE` (used as a component in `CS` [1]), is to split the budget of M records equally among all strata, so that each stratum receives M/r samples. While this is easy to implement and improves upon uniform sampling, this solution has the following drawback. Consider two groups 1 and 2 with the same means $\mu_1 = \mu_2$, but with very different standard deviations within the groups, i.e. $\sigma_1 \gg \sigma_2$. Intuitively, it is useful to give more samples to group 1 than to group 2, to reduce the variance of the estimate within group 1. However, `SENATE` gives the same number of samples to both, due to which the expected quality of the estimate for group 1 will be much worse than the estimate for group 2. Intuitively, we need to give more samples to group 1 than group 2, but exactly how much more – this is answered using our optimization framework.

Before proceeding further, we present a solution to an optimization problem that is repeatedly used in our work.

Lemma 1. *Consider positive valued variables $s_1, s_2, \ldots, s_k$ and positive constants $M, \alpha_1, \alpha_2, \ldots, \alpha_k$. The solution to the optimization problem: minimize $\sum_{i=1}^{k} \frac{\alpha_i}{s_i}$ subject to $\sum_{i=1}^{k} s_i \leq M$ is given by $s_i = M \cdot \frac{\sqrt{\alpha_i}}{\sum_{j=1}^{k} \sqrt{\alpha_j}}$.*

Proof. Let $s = [s_1, s_2, \ldots, s_k]$. Let $f(s) = \sum_{i=1}^{k} \frac{\alpha_1}{s_i}$, and $g(s) = (\sum_{i=1}^{k} s_i) - M$. We want to minimize $f(s)$ subject to the constraint $g(s) = 0$. Using Lagrange multipliers:

$$\begin{aligned} L(s_1, s_2, ..., s_k, \lambda) &= f(s_1, s_2, ..., s_k) + \lambda g(s_1, s_2, ..., s_k) \\ &= \sum_{i=1}^{k} \frac{\alpha_i}{s_i} + \lambda \left(\sum_{i=1}^{r} s_i - M \right) \end{aligned}$$

For each $i = 1 \ldots r$, we set $\frac{\partial L}{\partial s_i} = 0$. Thus $-\frac{\alpha_i}{s_i^2} + \lambda = 0$, leading to $s_i = \frac{\sqrt{\alpha_i}}{\sqrt{\lambda}}$. By setting $g(s) = 0$, we solve for λ and get $s_i = M \cdot \frac{\sqrt{\alpha_i}}{\sum_{j=1}^{k} \sqrt{\alpha_j}}$. □

We now consider how to find the best assignment of sample sizes to the different strata, using an optimization framework. Let $s = [s_1, s_2, \ldots, s_r]$ denote the vector of assignments of sample sizes to different strata.

Theorem 1. *For a single aggregation and single group-by, given weight vector w and sample size M, the optimal assignment of sample sizes is to assign to group $i \in \{1, 2, \ldots, r\}$ a sample size $s_i = M \frac{\sqrt{w_i}\sigma_i/\mu_i}{\sum_{j=1}^{r} \sqrt{w_j}\sigma_j/\mu_j}$*

Proof. Consider the estimators $y = [y_1, y_2, \ldots, y_r]$ computed using the sample. Our objective is the ℓ_2 error, which requires us to minimize $\sqrt{\sum_{i=1}^{r} w_i \cdot (\mathbb{CV}[y_i])^2}$, which is equivalent to minimizing $\sum_{i=1}^{r} w_i \cdot (\mathbb{CV}[y_i])^2$.

Algorithm 1: `CVOPT-SASG`: Algorithm computing a random sample for a single aggregate, single group-by.

Input: Database Table T, group-by attributes A, aggregation attribute d, weight vector w, memory budget M.

Output: Stratified Random Sample S

Let $\mathcal{A}$ denote all possibilities of assignments to A that actually occur in T. i.e. all strata. Let r denote the size of $\mathcal{A}$, and suppose the strata are numbered from 1 till r

For each $i = 1 \dots r$, compute the mean and variance of all elements in stratum i along attribute d, denoted as μ_i, σ_i respectively. Let $\gamma_i \leftarrow \sqrt{w_i}\sigma_i/\mu_i$

$\gamma \leftarrow \sum_{i=1}^{r} \gamma_i$

for $i = 1 \dots r$ **do**

  $s_i \leftarrow M \cdot \gamma_i / \gamma$

  Let S_i be formed by choosing s_i elements from stratum i uniformly without replacement, using reservoir sampling

return $S = [S_1, S_2, \dots, S_r]$

The standard deviation of y_i depends on n_i, the size of the ith group, s_i, the size of the sample assigned to the ith group, and σ_i, the standard deviation of the values in the ith group. By standard results on sampling, e.g. Theorem 2.2 in [8], we have

$$\mathbb{S}[y_i] = \sqrt{\frac{\sigma_i^2(n_i - s_i)}{n_i s_i}}, \mathbb{CV}[y_i] = \frac{1}{\mu_i}\sqrt{\frac{\sigma_i^2(n_i - s_i)}{n_i s_i}}$$

Thus, we reduce the problem to minimizing $\sum_{i=1}^{r} \frac{w_i}{\mu_i^2} \cdot \frac{\sigma_i^2(n_i - s_i)}{n_i s_i}$. Since r, σ_i, μ_i, and n_i are fixed, this is equivalent to minimizing $\sum_{i=1}^{r} \frac{w_i \sigma_i^2/\mu_i^2}{s_i}$ subject to the condition $\sum_{i=1}^{r} s_i = M$. Using Lemma 1, and setting $\alpha_i = w_i\sigma_i^2/\mu_i^2$, we see that s_i should be proportional to $\sqrt{w_i}\sigma_i/\mu_i$. □

The above leads to algorithm `CVOPT-SASG` for drawing a random sample from a table T, described in Algorithm 1.

3.2 Multiple Aggregates, Single Group-by

We next consider the case of multiple aggregations using the same group-by clause. Without loss of generality, suppose the columns that were aggregated are columns $1, 2, \dots, t$. As before, suppose the groups are numbered $1, 2, \dots, r$. For group $i, 1 \le i \le r$ and aggregation column $j, 1 \le j \le t$, let $\mu_{i,j}, \sigma_{i,j}$ respectively denote the mean and standard deviation of the values in column j within group i. Let n_i denote the size of group i, and s_i the number of samples drawn from group i. Let $y_{i,j}$ denote the estimate of $\mu_{i,j}$ obtained through the sample, and $\mathbb{CV}[y_{i,j}] = \frac{\mathbb{S}[y_{i,j}]}{\mu_{i,j}}$ denote the coefficient of variation of $y_{i,j}$. Further suppose that we are given weights for each combination of group and aggregate, which reflect how important these are to the user [3]. Let $w_{i,j}$ denote the weight of the combination group i and aggregation column j. Our minimization metric is a weighted combination of the coefficients of variation of all the $r \cdot t$ estimates, one for each group

[3] In the absence of user-input weights, we can assume default weights of 1.

and aggregate combination.

$$\text{error}(y, w) = \sqrt{\sum_{j=1}^{t} \sum_{i=1}^{r} w_{i,j} \cdot (\mathbb{CV}[y_{i,j}])^2}$$

Theorem 2. *Given weights w, and total sample size M, the optimal assignment of sample sizes s among r groups is to assign to group $i = 1, 2, \ldots, r$ sample size $s_i = M \frac{\sqrt{\alpha_i}}{\sum_{i=1}^{r} \sqrt{\alpha_i}}$, where $\alpha_i = \sum_{j=1}^{t} \frac{w_{i,j} \sigma_{i,j}^2}{\mu_{i,j}^2}$.*

Proof. We use an approach similar to Theorem 1. Let $y = \{y_{i,j}\} 1 \leq i \leq r, 1 \leq j \leq t$ denote the matrix of estimators for the means of the multiple aggregates, for different groups. Using the metric of ℓ_2 error, we have to minimize $\ell_2(\mathbb{CV}[y], w) = \sqrt{\sum_{i=1}^{r} \sum_{j=1}^{t} w_{i,j} \cdot (\mathbb{CV}[y_{i,j}])^2}$, which is equivalent to minimizing $\sum_{i=1}^{r} \sum_{j=1}^{t} w_{i,j} \cdot \left(\frac{\mathbb{S}[y_{i,j}]}{\mu_{i,j}} \right)^2$.

We note that $\mathbb{S}[y_{i,j}] = \sqrt{\frac{\sigma_{i,j}^2 (n_i - s_i)}{n_i s_i}}$, where $\sigma_{i,j}$ is the standard deviation of the jth column taken over all elements in group i. Thus, we reduce the problem to minimizing $\sum_{i=1}^{r} \sum_{j=1}^{t} w_{i,j} \cdot \frac{\sigma_{i,j}^2 (n_i - s_i)}{n_i s_i} \cdot \frac{1}{\mu_{i,j}^2}$. Since r, t, $\sigma_{i,j}$, $\mu_{i,j}$, and n_i are fixed, this is equivalent to minimizing:

$$\sum_{i=1}^{r} \sum_{j=1}^{t} w_{i,j} \frac{\sigma_{i,j}^2}{\mu_{i,j}^2 s_i} = \sum_{i=1}^{r} \frac{1}{s_i} \sum_{j=1}^{t} \frac{w_{i,j} \sigma_{i,j}^2}{\mu_{i,j}^2} = \sum_{i=1}^{r} \frac{\alpha_i}{s_i} \tag{1}$$

subject to the condition $\sum_{i=1}^{r} s_i \leq M$. Using Lemma 1, we arrive that the optimal assignment is $s_i = M \frac{\sqrt{\alpha_i}}{\sum_{i=1}^{r} \sqrt{\alpha_i}}$. □

In our formulation, a weight can be assigned to each result in the output, reflecting how important this number is. For instance, if there are r groups and t aggregates desired, then there are $r \times t$ results in the output, and the user can assign a weight for each result, w_{ij} for $i = 1 \ldots r$ and $j = 1 \ldots t$. A useful special case is when all weights are equal, so that all results are equally important to the user. If the user desires greater accuracy for group 1 when compared to group 2, this can be done by setting weights $w_{1,*}$ to be higher than the weights $w_{2,*}$, say 10 versus 1. The value of weight can also be deduced from a query workload, as we discuss in Section 4.3.

4 Multiple Group-Bys

Suppose that we had multiple attribute combinations on which there are group-by clauses. For instance, we may have a query where the student data is being grouped by `major`, and one query where it is being grouped by `year`, and another query where data is grouped by `major` as well as `year`. The additional challenge now is that there are multiple ways to stratify the data, to further apply stratified sampling. For instance, we can draw a stratified sample where data are stratified according to `major` only, or one where data are stratified according to `year`, or one where data are stratified according to both `major` and `year`. Any of these three samples can be used to answer all three queries, but may lead to high errors. For instance, a stratified sample where data is stratified

according to year of graduation may lead to poor estimates for a group-by query based on `major`, since it may yield very few tuples or may completely miss some majors.

Our solution is to pursue a "finest stratification" approach where the population is stratified according to the union of all group-by attributes. In the above example, this leads to stratification according to a combination of `major` and `year`, leading to one stratum for each distinct value of the pair `(year,major)`. This will serve group-by queries based solely on `major` or `year`, or a combination of both. The number of samples assigned to each stratum in such a stratification is determined in a principled manner.

4.1 Single Aggregate, Multiple Group-By

We first consider the case of a single aggregate and multiple group-bys, starting with the case of two group-bys and then extend to more than two group-bys. Suppose two queries Q_1 and Q_2 that aggregate on the same column, using different sets of group-by attributes, A and B, respectively. Note that A and B need not be disjoint. For example A can be (`major`, `year`) and B can be (`major`, `zipcode`). If we combined the sets of group-by attributes, we get attribute set $C = A \cup B$. In the above example, C is (`major`, `year`, `zipcode`). Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ denote the set of all values possible for attributes in A, B, and C respectively. Note that only those combinations that actually occur within data are considered.

Our algorithm based on finest stratification stratifies data according to attribute set C, leading to a stratum for each combination of the values of attributes $c \in \mathcal{C}$. Samples are chosen uniformly within a single stratum, but the sampling probabilities in different strata may be different. *Our goal is* **not** *to get a high-quality estimate for aggregates within each stratum according to $\mathcal{C}$. Instead, it is to get a high-quality estimate for aggregates for each group in $\mathcal{A}$ (query Q_1) and in $\mathcal{B}$ (query Q_2).* We translate the above goal into an objective function that will help assign sample sizes to each stratum in $\mathcal{C}$.

For each stratum $c \in \mathcal{C}$, let s_c denote the number of samples assigned to this stratum, S_c the sample, μ_c the mean of the aggregation column, and σ_c the standard deviation of the aggregation column. Let the sample mean for this stratum be denoted as $y_c = \frac{\sum_{v \in S_c} v}{s_c}$. As $C = A \cup B$, $A \in C$. For an assignment $a \in \mathcal{A}$ and an assignment $c \in \mathcal{C}$, we say $c \in a$ if the attributes in set A have the same values in a and c. Let $\mathcal{C}(a)$ denote the set of all $c \in \mathcal{C}$ such that $c \in a$. For any $c \in \mathcal{C}$, let $\Pi(c, A)$ denote the unique $a \in \mathcal{A}$ such that $c \in \mathcal{C}(a)$. Similarly, define $\Pi(c, B)$.

For query Q_1, for group $a \in \mathcal{A}$, let μ_a denote the mean of aggregate column, and n_a denote the size of the group. We desire to estimate μ_a for each $a \in \mathcal{A}$. Suppose the estimate for μ_a is y_a. Similarly, we define μ_b, n_b, and y_b for each group $b \in \mathcal{B}$. Our objective function is the weighted ℓ_2 norm of $\{\mathbb{CV}[y_a] \,|a \in \mathcal{A}\} \cup \{\mathbb{CV}[y_b] \,|b \in \mathcal{B}\}$, i.e.

$$\sqrt{\sum_{a \in \mathcal{A}} w_a \cdot (\mathbb{CV}[y_a])^2 + \sum_{b \in \mathcal{B}} w_b \cdot (\mathbb{CV}[y_b])^2}$$

The estimates for each group are derived as $y_a = \frac{\sum_{c \in \mathcal{C}(a)} n_c y_c}{\sum_{c \in \mathcal{C}(a)} n_c}$, and similarly $y_b = \frac{\sum_{c \in \mathcal{C}(b)} n_c y_c}{\sum_{c \in \mathcal{C}(b)} n_c}$. Using standard results from stratified sampling [8], we have: $\mathbb{E}[y_a] = \mu_a$, and

$$\mathbb{VAR}[y_a] = \frac{1}{n_a^2} \sum_{c \in \mathcal{C}(a)} \left[\frac{n_c^2 \sigma_c^2}{s_c} - n_c \sigma_c^2 \right]$$

Lemma 2. *The optimal assignment of sample sizes that minimizes the weighted ℓ_2 norm of the coefficients of variation of the estimates is: for $d \in \mathcal{C}$ the sample size is $s_d = M \cdot \frac{\sqrt{\beta_d}}{\sum_{c\in\mathcal{C}} \sqrt{\beta_c}}$, where:*

$$\beta_c = n_c^2 \sigma_c^2 \left[\frac{w_{\Pi(c,A)}}{n_{\Pi(c,A)}^2 \mu_{\Pi(c,A)}^2} + \frac{w_{\Pi(c,B)}}{n_{\Pi(c,B)}^2 \mu_{\Pi(c,B)}^2} \right]$$

Proof. Our objective function is the weighted ℓ_2 norm of the coefficients of variance of all estimators $\{y_a | a \in \mathcal{A}\}$ and $\{y_b | b \in \mathcal{B}\}$, which we want to minimize over all possibilities of the sample sizes $s = \{s_c | c \in \mathcal{C}\}$, subject to $\sum_{c\in\mathcal{C}} s_c = M$. Equivalently, we minimize the square of the weighted ℓ_2 norm:

$$\begin{aligned} Y(s) &= \sum_{a\in\mathcal{A}} w_a \cdot (\mathbb{CV}[y_a])^2 + \sum_{b\in\mathcal{B}} w_b \cdot (\mathbb{CV}[y_b])^2 \\ &= \sum_{a\in\mathcal{A}} \frac{w_a \cdot \mathbb{VAR}[y_a]}{\mu_a^2} + \sum_{b\in\mathcal{B}} \frac{w_b \cdot \mathbb{VAR}[y_b]}{\mu_b^2} \end{aligned}$$

Using the exp. and variance of y_a, we can rewrite $Y(s)$:

$$\begin{aligned} Y(s) = &\sum_{a\in\mathcal{A}} \frac{w_a \cdot \sum_{c\in\mathcal{C}(a)} \left[\frac{n_c^2\sigma_c^2}{s_c} - n_c\sigma_c^2 \right]}{n_a^2\mu_a^2} \\ &+ \sum_{b\in\mathcal{B}} \frac{w_b \cdot \sum_{c\in\mathcal{C}(b)} \left[\frac{n_c^2\sigma_c^2}{s_c} - n_c\sigma_c^2 \right]}{n_b^2\mu_b^2} \end{aligned}$$

This is equivalent to minimizing:

$$\begin{aligned} Y'(s) &= \sum_{a\in\mathcal{A}} \sum_{c\in\mathcal{C}(a)} \frac{w_a n_c^2 \sigma_c^2}{s_c n_a^2 \mu_a^2} + \sum_{b\in\mathcal{B}} \sum_{c\in\mathcal{C}(b)} \frac{w_b n_c^2 \sigma_c^2}{s_c n_b^2 \mu_b^2} \\ &= \sum_{c\in\mathcal{C}} \left[\frac{w_{\Pi(c,A)} n_c^2 \sigma_c^2}{s_c n_{\Pi(c,A)}^2 \mu_{\Pi(c,A)}^2} + \frac{w_{\Pi(c,B)} n_c^2 \sigma_c^2}{s_c n_{\Pi(c,B)}^2 \mu_{\Pi(c,B)}^2} \right] \end{aligned}$$

We note that the problem turns to: minimize $Y'(s) = \sum_{c\in\mathcal{C}} \frac{\beta_c}{s_c}$ subject to $\sum_{c\in\mathcal{C}} s_c = M$. Using Lemma 1, we arrive that the optimal assignment of sample size is $s_c = M \cdot \frac{\sqrt{\beta_c}}{\sum_{d\in\mathcal{C}} \sqrt{\beta_d}}$. □

An example: Consider a query Q_1 that groups by `major` and aggregates by `gpa`, and another query Q_2 that groups by `year`, and aggregates by `gpa`. Suppose each group in each query has the same weight 1. The above algorithm stratifies according to the `(major,year)` combination. For a stratum where `major` equals m and `year` equals y, sample size $s_{m,y}$ is proportional to:

$$\beta_{m,y} = n_{m,y}^2 \sigma_{m,y}^2 \left[\frac{1}{n_{m,*}^2 \mu_{m,*}^2} + \frac{1}{n_{*,y}^2 \mu_{*,y}^2} \right]$$

Where $n_{m,y}, n_{m,*}, n_{*,y}$ are respectively the number of elements with `major` $= m$ and `year` $= y$, the number of elements with `major` $= m$, and the number of elements with `year` $= y$, respectively. Similarly for $\mu_{m,y}, \mu_{m,*}, \mu_{*,y}$.

Example 2: Consider a query R_1 that groups by `major, year` and aggregates by `gpa`, and another query R_2 that groups by `zipcode, year`, and aggregates by `gpa`. Suppose all groups in both queries share the same weight 1. The above algorithm asks to stratify according to `(major, zipcode, year)` combination. For a stratum where `major` equals m, `zipcode` equals z and `year` equals y, sample size $s_{m,z,y}$ is proportional to:

$$\beta_{m,z,y} = n_{m,z,y}^2 \sigma_{m,z,y}^2 \left[\frac{1}{n_{m,*,y}^2 \mu_{m,*,y}^2} + \frac{1}{n_{*,z,y}^2 \mu_{*,z,y}^2} \right]$$

Where $n_{m,z,y}$, $n_{m,*,y}$, and $n_{*,z,y}$ are respectively the number of elements with `major` equal to m and `zipcode` equal to z and `year` equal to y, the number of elements with `major` equal to m and `year` equal to y, and the number of elements with `zipcode` equal to z and `year` equal to y, respectively. Similarly for $\mu_{m,z,y}, \mu_{m,*,y}, \mu_{*,z,y}$.

Generalizing to Multiple Group-Bys Suppose there were multiple group-by queries with attribute sets $A_1, A_2, \ldots, A_k$. The algorithm stratifies according to attribute set $C = \bigcup_{i=1}^{k} A_i$. For $i = 1 \ldots k$, let $\mathcal{A}_i$ denote the universe of all assignments to attributes in A_i and $\mathcal{C}$ the universe of all possible assignments to attributes in C. Note that only those assignments that exist in the data need be considered. Extending the above analysis for the case of two group-bys, we get that the optimal assignment of samples as follows. For each $c \in \mathcal{C}$, stratum c is assigned sample size proportional to the square root of:

$$\beta_c = n_c^2 \sigma_c^2 \sum_{i=1}^{k} \frac{w_{\Pi(c,A_i)}}{n_{\Pi(c,A_i)}^2 \mu_{\Pi(c,A_i)}^2}$$

The proof of above result is similar to the case of Lemma 2. We minimize the ℓ_2 norm:

$$Y(s) = \sum_{a_i \in \mathcal{A}_i} \frac{w_{a_i} \cdot \sum_{c \in \mathcal{C}(a_i)} \left[\frac{n_c^2 \sigma_c^2}{s_c} - n_c \sigma_c^2 \right]}{n_{a_i}^2 \mu_{a_i}^2}$$

This is equivalent to minimizing:

$$\begin{aligned} Y'(s) &= \sum_{a_i \in \mathcal{A}_i} \sum_{c \in \mathcal{C}(a_i)} \frac{w_{a_i} n_c^2 \sigma_c^2}{s_c n_{a_i}^2 \mu_{a_i}^2} \\ &= \sum_{c \in \mathcal{C}} \sum_{i=1}^{k} \frac{w_{\Pi(c,A_i)} n_c^2 \sigma_c^2}{s_c n_{\Pi(c,A_i)}^2 \mu_{\Pi(c,A_i)}^2} = \sum_{c \in \mathcal{C}} \frac{\beta_c}{s_c} \end{aligned}$$

Using Lemma 1, we have the result proved.

Cube-By Queries An important special case of multiple group-by queries, often used in data warehousing, is the *cube-by* query. The cube-by query takes as input a set of attributes A and computes group-by aggregations based on the entire set A as well as every subset of A. For instance, if A was the set `major, year` and the aggregation column is A, then the cube-by query poses four queries, one grouped by `major` and `year`, one grouped by only `major`, one grouped by only `year`, and the other without a group-by (i.e. a full table query). Our algorithm for multiple group-by can easily handle the case of a cube-by query and produce an allocation that optimizes the ℓ_2 norm of the CVs of all estimates. We present an experimental study of cube-by queries in Section 6.

4.2 Multiple Aggregates, Multiple Group-Bys

Suppose two queries, Q_1, Q_2, that aggregate on the different columns d_1 and d_2 and also use different sets of group-by attributes, A and B that may be overlapping. *e.g.,* Q_1 can aggregate `gpa` grouped by (`major`, `year`) and Q_2 can aggregate `credits` grouped by (`major`, `zipcode`).

We stratify the data according to attribute set $C = A \cup B$. As in Section 4.1, let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ denote the set of all values possible for attributes in A, B, and C respectively. Also, for $a \in \mathcal{A}$, $b \in \mathcal{B}$, $c \in \mathcal{C}$, let $\mathcal{C}(a)$, $\mathcal{C}(b)$, $\Pi(c, A)$ and $\Pi(c, B)$ be defined as in Section 4.1.

For each $c \in \mathcal{C}$, let n_c denote the number of data elements in this stratum, $\sigma_{c,1}$ the variance of the d_1 column among all elements in this stratum, and $\sigma_{c,2}$ the variance of the d_2 column in this stratum. Let s_c denote the number of samples assigned to this stratum, and $y_{c,1}$ and $y_{c,2}$ respectively denote the sample means of the columns d_1 and d_2 among all elements in stratum c respectively.

For each $a \in \mathcal{A}$, we seek to estimate $\mu_{a,1}$, the mean of the d_1 column among all elements in this group. The estimate, which we denote by $y_{a,1}$ is computed as $\frac{\sum_{c \in \mathcal{C}(a)} n_c y_{c,1}}{\sum_{c \in \mathcal{C}(a)} n_c}$. Similarly for each $b \in \mathcal{B}$, we seek to estimate $\mu_{b,2}$ the mean of the d_2 column among all elements in this group. Let $y_{b,2}$ be this estimate. Our optimization metric is the weighted ℓ_2 norm of the coefficients of variation of all estimates:

$$\mathcal{L} = \sum_{a \in \mathcal{A}} w_{a,1} \cdot (\mathbb{CV}[y_{a,1}])^2 + \sum_{b \in \mathcal{B}} w_{b,2} \cdot (\mathbb{CV}[y_{b,2}])^2$$

Lemma 3. *For two group-by and two aggregates, the optimal assignment of sample sizes that minimizes the weighted ℓ_2 norm of the coefficients of variation of the estimates is: for $d \in \mathcal{C}$ the sample size is $s_d = M \cdot \frac{\sqrt{\beta_d}}{\sum_{c \in \mathcal{C}} \sqrt{\beta_c}}$, where*

$$\beta_c = n_c^2 \left[\frac{w_{\Pi(c,A),1} \sigma_{c,1}^2}{n_{\Pi(c,A)}^2 \mu_{\Pi(c,A),1}^2} + \frac{w_{\Pi(c,B),2} \sigma_{c,2}^2}{n_{\Pi(c,B)}^2 \mu_{\Pi(c,B),2}^2} \right]$$

Proof. Our objective function is the weighted ℓ_2 norm of the coefficients of variance of all estimators $\{y_{a,1} | a \in \mathcal{A}\}$ and $\{y_{b,2} | b \in \mathcal{B}\}$, which we want to minimize over all possibilities of the vector of sample sizes $s = \{s_c | c \in \mathcal{C}\}$, subject to $\sum_{c \in \mathcal{C}} s_c = M$. Equivalently, we minimize the square of the weighted ℓ_2 norm:

$$\begin{aligned} Y(s) &= \sum_{a \in \mathcal{A}} w_{a,1} \cdot (\mathbb{CV}[y_{a,1}])^2 + \sum_{b \in \mathcal{B}} w_{b,2} \cdot (\mathbb{CV}[y_{b,2}])^2 \\ &= \sum_{a \in \mathcal{A}} \frac{w_{a,1} \cdot \mathbb{VAR}[y_{a,1}]}{\mu_{a,1}^2} + \sum_{b \in \mathcal{B}} \frac{w_{b,2} \cdot \mathbb{VAR}[y_{b,2}]}{\mu_{b,2}^2} \end{aligned}$$

Using the expected value and variance of y_a, we can rewrite $Y(s)$:

$$Y(s) = \sum_{a \in \mathcal{A}} \frac{w_{a,1} \cdot \sum_{c \in \mathcal{C}(a)} \left[\frac{n_c^2 \sigma_{c,1}^2}{s_c} - n_c \sigma_{c,1}^2 \right]}{n_a^2 \mu_{a,1}^2} + \sum_{b \in \mathcal{B}} \frac{w_{b,2} \cdot \sum_{c \in \mathcal{C}(b)} \left[\frac{n_c^2 \sigma_{c,2}^2}{s_c} - n_c \sigma_{c,2}^2 \right]}{n_b^2 \mu_{b,2}^2}$$

This is equivalent to minimizing:

$$\begin{aligned} Y'(s) &= \sum_{a\in\mathcal{A}} \sum_{c\in\mathcal{C}(a)} \frac{w_{a,1} n_c^2 \sigma_{c,1}^2}{s_c n_a^2 \mu_{a,1}^2} + \sum_{b\in\mathcal{B}} \sum_{c\in\mathcal{C}(b)} \frac{w_b n_c^2 \sigma_{c,2}^2}{s_c n_b^2 \mu_{b,2}^2} \\ &= \sum_{c\in\mathcal{C}} \left[\frac{w_{\Pi(c,A)} n_c^2 \sigma_{c,1}^2}{s_c n_{\Pi(c,A)}^2 \mu_{\Pi(c,A),1}^2} + \frac{w_{\Pi(c,B)} n_c^2 \sigma_{c,2}^2}{s_c n_{\Pi(c,B)}^2 \mu_{\Pi(c,B),2}^2} \right] \\ &= \sum_{c\in\mathcal{C}} \frac{\beta_c}{s_c} \end{aligned}$$

Subject to $\sum_{c\in\mathcal{C}} s_c = M$. Using Lemma 1, we arrive that the optimal assignment of sample size is $s_c = M \cdot \frac{\sqrt{\beta_c}}{\sum_{d\in\mathcal{C}} \sqrt{\beta_d}}$. □

We can generalize this to the case of more than two aggregations, and/or more than two group-bys. Suppose there were k group-by queries $Q_1, Q_2, \ldots, Q_k$, with attribute sets $A_1, A_2, \ldots, A_k$. Each query Q_i has multiple aggregates on a set of columns denoted as $\mathcal{L}_i$. In this case, the algorithm stratifies according to attribute set $C = \bigcup_{i=1}^{k} A_i$. For $i = 1 \ldots k$, let $\mathcal{A}_i$ denote the universe of all assignments to attributes in A_i and $\mathcal{C}$ the universe of all possible assignments to attributes in C. Note that only those assignments that exist in the data need be considered. Extending the analysis from Section 3.2, 4.1, and 4.2, we get that the optimal assignment of samples is as follows. For each $c \in \mathcal{C}$, stratum c is assigned sample size proportional to the square root of

$$\beta_c = n_c^2 \sum_{i=1}^{k} \left(\frac{1}{n_{\Pi(c,A_i)}^2} \sum_{\ell\in\mathcal{L}_i} \frac{w_{\Pi(c,A_i),\ell} \cdot \sigma_{c,\ell}^2}{\mu_{\Pi(c,A_i),\ell}^2} \right).$$

4.3 Using A Query Workload

How can one use (partial) knowledge of a query workload to improve sampling? A query workload is a probability distribution of expected queries, and can be either collected from historical logs or created by users based on their experience. In the presence of a query workload, we show how to construct a sample that is optimized for this workload. We focus on the case of multiple aggregations, multiple group-by (MAMG), as others such as SASG and MASG are special cases. Our approach is to use the query workload to deduce the weights that we use in the weighted optimization for group-by queries.

An Example: Consider an example `Student` table and its query workload shown in Tables 1 and 2. The workload has 45 group-by queries, of which three are distinct, named A, B, and C. Each group-by query stratifies its aggregation columns into a collection of mutually disjoint *aggregation groups*. Each aggregation group is identified by a tuple of (a, b), where a is the aggregation column name and b is one value assignment to the group-by attributes. For example, Query A stratifies the `age` column into four aggregation groups: *(age,major=CS)*, *(age, major=Math)*, *(age,major=EE)*, and *(age,major=ME)*. Each aggregation group is a subset of elements in the aggregation column, *e.g.,* aggregation group *(age,major=CS)* is the set $\{25, 22\}$. One aggregation group may appear more than once, because one query may occur multiple times and different queries may share aggregation group(s). Our preprocessing is to deduce all the aggregation groups and their frequencies from the workload. Table 3 shows the result of the example workload. We then use each aggregation group's frequency as its weight in the optimization framework for `CVOPT` sampling.

Table 1: An example `Student` table

id	age	GPA	SAT	major	college
1	25	3.4	1250	CS	Science
2	22	3.1	1280	CS	Science
3	24	3.8	1230	Math	Science
4	28	3.6	1270	Math	Science
5	21	3.5	1210	EE	Engineering
6	23	3.2	1260	EE	Engineering
7	27	3.7	1220	ME	Engineering
8	26	3.3	1230	ME	Engineering

Table 2: An example query workload on the `Student` table

Queries	repeats
A: `SELECT AVG(age), AVG(gpa) FROM Student GROUP BY major`	20
B: `SELECT AVG(age), AVG(sat) FROM Student GROUP BY college`	10
C: `SELECT AVG(gpa) FROM Student GROUP BY major WHERE college=Science`	15

5 CVOPT-INF and Extensions

We now consider minimizing for the maximum of the CVs, or the ℓ_∞ norm of all $\mathbb{CV}[y_i]$ for different groups i. That is:

$$\ell_\infty(\mathbb{C}) = \max_{i=1}^{r} \mathbb{CV}[y_i] = \max_{i=1}^{r} \frac{\sigma_i}{\mu_i}\sqrt{\frac{n_i - s_i}{n_i s_i}}$$

subject to the constraint $\sum_{i=1}^{r} s_i \le M$. One obvious benefit of using $\ell_\infty(\mathbb{C})$ as the objective function is that the relative errors of different groups are expected to be the same.

The above problem has integrality constraints and is hard to optimize exactly. In the rest of this section, we present an efficient algorithm that relaxes the integrality constraints and assumes that s_is can have real values. Note that we assume every $\sigma_i > 0$, any group i where $\sigma_i = 0$ can be treated as a special case, since all its values are equal, and there is no need to maintain a sample of that group.

An efficient algorithm: Consider a collection of r functions

$$f_i(x) = \frac{\sigma_i}{\mu_i}\sqrt{\frac{n_i - x}{n_i x}}, \quad i = 1, 2, \ldots, r$$

where $x \in [0, M]$ is a real number. Our goal is find an assignment of $x = x_i$ for each f_i, that minimizes

$$\max_{i=1}^{r} f_i(x_i) = \max_{i=1}^{r} \frac{\sigma_i}{\mu_i}\sqrt{\frac{n_i - x_i}{n_i x_i}}$$

and $\sum_{i=1}^{r} x_i \le M$. Let $x_1^*, x_2^*, \ldots, x_r^*$ denote such assignments that solve this continous optimization problem.

Table 3: Aggregation groups and their frequencies produced from the example workload (Table 2)

Aggregation groups	Frequency
(age, major=CS), (age, major=Math), (age, major=EE) (age, major=ME), (GPA, major=EE), (GPA, major=ME)	25
(GPA, major=CS), (GPA, major=Math)	35
(age, college=Science), (age, college=Engineering) (SAT, college=Science), (SAT, college=Engineering)	10

Lemma 4. $f_1(x_1^*) = f_2(x_2^*) = \ldots = f_r(x_r^*)$

Proof. We use proof by contradiction. Suppose the claim in the lemma is not true, then without losing generality, say $f_1(x_1^*), f_k(x_k^*), \ldots, f_k(x_k^*)$, for some $k < r$, are all the largest and $f_r(x_r^*)$ is the smallest. Then, because each f_i is a strictly decreasing function, we can always reduce the value of each x_i^*, $i \leq k$, for some amount c and increase the value of x_r^* by amount kc, such that (1) every f_i is decreased, $i \leq k$. (2) f_r is increased, and (3) $\max_{i=1}^r f_i$ is decreased. This is a contradiction, since we already know $\max_{i=1}^r f_i(x_i^*)$ was minimized. □

Following Lemma 4, we can have

$$\frac{\sigma_1}{\mu_1}\sqrt{\frac{n_1 - x_1^*}{n_1 x_1^*}} = \frac{\sigma_2}{\mu_2}\sqrt{\frac{n_2 - x_2^*}{n_2 x_2^*}} = \ldots \frac{\sigma_r}{\mu_r}\sqrt{\frac{n_r - x_r^*}{n_r x_r^*}}$$

$$\implies \frac{d_1}{x_1^*/\bar{x}_1^*} = \frac{d_2}{x_2^*/\bar{x}_2^*} = \ldots = \frac{d_r}{x_r^*/\bar{x}_r^*} \tag{2}$$

where $d_i = \frac{(\sigma_i/\mu_i)^2}{n_i}$ and $\bar{x}_i^* = n_i - x_i^*$. Let $D = \sum_{i=1}^r d_i$. By Equations 2, each $x_i^*/\bar{x}_i^*$ is proportional to d_i, i.e, $x_i^*/\bar{x}_i^* = q^* \cdot d_i/D$ for some real number constant $q^* \in [0, n]$. Namely,

$$x_i^* = \frac{q^* \cdot d_i/D}{1 + q^* \cdot d_i/D} n_i.$$

Our approach to minimize $\ell_\infty(\mathbb{C})$ is to perform a binary search for the largest integer $q \in [0, n]$ that approximates q^*, in such

$$\sum_{i=1}^r x_i = \sum_{i=1}^r \frac{q \cdot d_i/D}{1 + q \cdot d_i/D} n_i \leq M.$$

If the binary search returns $q = 0$, we set $q = 1$. We then assign each $s_i = \left\lceil \frac{x_i}{\sum_{j=1}^r x_j} M \right\rceil$. Clearly, the total time cost for finding the set of s_i values is $O(r \log n)$: There are a total of $O(\log n)$ binary search steps where each step takes $O(r)$ time.

Extension to Other Aggregates. Thus far, our discussion of the `CVOPT` framework has focused on group-by queries using the AVG aggregates (COUNT and SUM are very similar). `CVOPT` can be extended to other aggregates as well. To use the framework for an aggregate, we need to: (1) have the per-group CV of the aggregate of interest well defined, and (2) ensure that it is possible to compute the CV of a stratum using statistics stored for strata in finer stratification of this stratum. Hence, the method can potentially be extended to aggregates such as per-group median and variance.

6 Experimental Evaluation

We evaluate CVOPT for approximate query processing using Hive [22] as the underlying data warehouse. The first phase is an offline sample computation that performs two passes of the data. The first pass computes some statistics for each group, and the second pass uses these statistics as input for CVOPT to decide the sample sizes for different groups and to draw the actual sample.

In the second phase, the sample obtained using CVOPT is used to (approximately) answer queries. The sample from CVOPT is representative and can answer queries that involve selection predicates provided at query time, as well as new combinations of groupings, without a need for recomputing the sample at query time. Overall, we can expect the overhead of the offline sampling phase to be small when compared with the time saved through sample-based query processing.

We collected two real-world datasets, OpenAQ and Bikes. OpenAQ [18] is a collection of the air quality measurements of different substances, such as carbon monoxide, sulphur dioxide, *etc.* The data consists of about 200 million records, collected daily from ten thousand locations in 67 countries from 2015 to 2018. Bikes is data logs from Divvy [24], a Chicago bike share company. Customers can pick up a bike from one Divvy station kiosk and return it to any station at their convenience. The dataset contains information about these bike rides and also has some user information, such as gender or birthday. We analyzed all subscribers' data from 2016 to 2018, for a total of approximately 11.5 million records. The datasets are stored in the database as 2 tables OpenAQ and Bike. Throughout this section, we introduce queries to those 2 tables, annotated with "AQ" and "B" prefixes, respectively.

Approximate answers from sample table are compared to the ground truth, derived using an exact computation from the full data. Let x and $\bar{x}$ be the ground-truth and approximate answer, respectively. We use the relative error $|\bar{x} - x|/x$ as the error metric for each group.

We implemented algorithms Uniform, Sample+Seek, CS and RL to compare with CVOPT (see Section 1.2 for a brief overview). Uniform is the unbiased sampler, which samples records uniformly without replacement from the base table. RL is the algorithm due to Rosch and Lehner [20]. CS is congressional sampling algorithm due to [1]. Sample+Seek is from [9], after applying appropriate normalization to get an unbiased answer. CVOPT is the implementation of our ℓ_2 optimal sampler. We also report results from CVOPT-INF, the ℓ_∞-optimal sampler. Unless otherwise specified, each method draws a 1% sample. Each reported result is the average of 5 identical and independent repetitions of an experiment.

6.1 Accuracy of Approximate Query Processing

The quality of a sample is measured by the accuracy of the approximate answers using the sample. We introduce MASG queries AQ1, AQ2 and B1. AQ1 is a relatively complex, realistic query computing the changes of both the average and the number of days with high level of black carbon *(bc)*, for each country between 2017 and 2018. The query contains different aggregates, multiple table scans, and a join operator. AQ2 and B1 are simpler examples of MASG query, which has multiple aggregate functions sharing the group-by.

Figure 1 shows the maximum errors of the approximate answers of query AQ1 using a 1% sample. We report the maximum error across all answers. CVOPT shows a significant improvement over other methods. It has a maximum error of 8.8% while CS and RL have error of as much as 50%. With the same space budget, the error of Uniform can be as large as 135%, as some groups are poorly represented. Similar improvements are observed with other MASG queries. For AQ2,

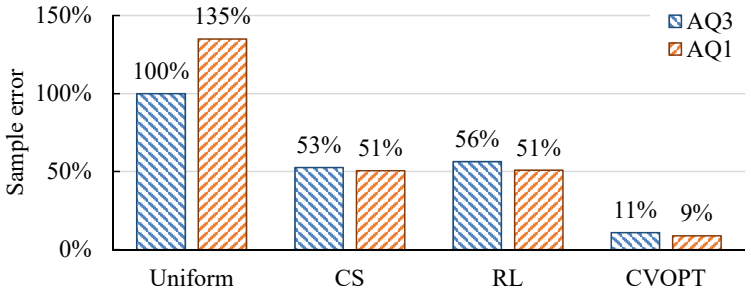


Figure 1: Maximum error for MASG query AQ1 and SASG query AQ3 using a 1% sample.

	OpenAQ				Bikes			
	SASG	MASG	SAMG	MAMG	SASG	MASG	SAMG	MAMG
Uniform	21.2	19.0	12.3	10.9	14.7	9.0	24.0	20.5
Sample+Seek	38.4	20.9	34.1	33.2	10.9	15.6	15.3	15.2
CS	2.1	1.1	3.2	2.3	4.8	2.6	6.9	5.2
RL	3.0	1.8	4.5	3.6	4.3	2.8	7.6	5.8
CVOPT	1.6	0.8	2.4	2.2	4.0	2.3	6.3	4.8

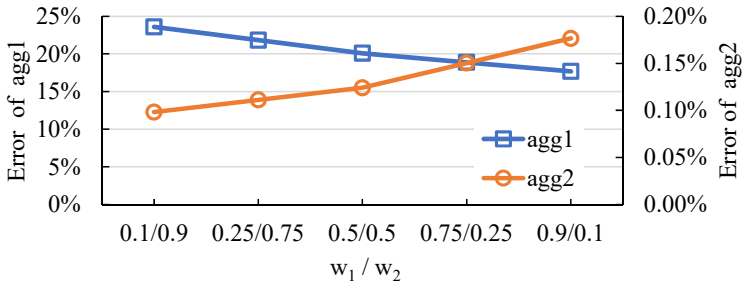
Table 4: Percentage average error for different queries, OpenAQ and Bikes datasets, with 1% and 5% samples, respectively.

the maximum errors of CS, RL and CVOPT are 10.1%, 29.5% and 5.9% respectively. For B1 the maximum errors of CS, RL and CVOPT are 11.7%, 8.8% and 7.7%, respectively.

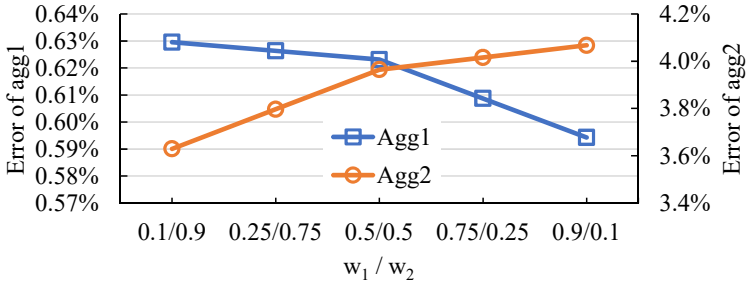
We present queries AQ3, B2 and AQ4 as case-studies for SASG query. AQ4 is a realistic example, while AQ3 and B2 are simpler examples. Figure 1 shows the maximum errors for AQ3 using a 1% sample. For both SASG and MASG queries, CVOPT yields the lowest error. While CVOPT has 11% sample error, CS and RL have large errors of more than 50%. For SASG query, *e.g.,* AQ4, CVOPT and RL share similar objective functions. However, RL assumes that the size of a group is always large, and in allocation sample sizes, does not take the group size into account (it only uses the CV of elements in the group). However real data, including the OpenAQ dataset, may contain small groups, where RL may allocate a sample size greater than the group size. CVOPT does not make such an assumption, and hence often leads to better quality samples than RL, even for the case of SASG.

Uniform has largest error of 100%, as some groups are absent in Uniform sample. Similar results are seen in other SASG queries, where CS, RL and CVOPT have the maximum errors 39%, 22% and 21% for B2; and 14%, 34% and 8% for AQ4, respectively.

Table 4 summarizes the average errors of different queries. Generally, CVOPT shows the best average error among different methods. In some cases, CVOPT has minor improvements for average error, but for maximum error, CVOPT significantly outperforms others. That is because CVOPT gives a good representation for *all groups* while others do not guarantee to cover all groups. The order of other methods changes for different queries, as each method has an advantageous query type, while CVOPT is fairly stable across multiple types of query. Note that Queries AQ3, B1, and B2 have selection predicates, which are applied after the sampling is performed. We study how the sample can be reused for predicates with different selectivities and for different group-by attributes



(a) 1% CVOPT sample answers query AQ2 with weight settings.



(b) 5% CVOPT sample answers query B1 with weight settings.

Figure 2: Average errors, CVOPT, for different weight settings.

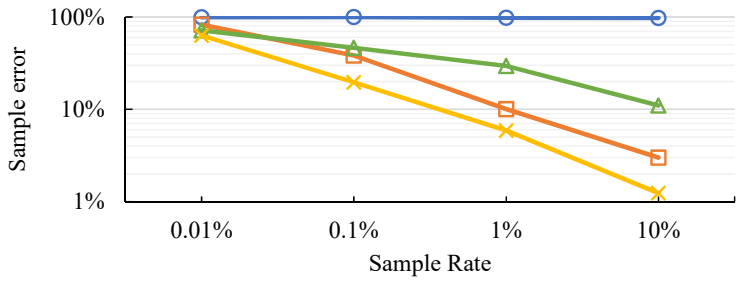
in Section 6.3. Since the error of Sample+Seek can be very large (maximum error as high as 173%), we exclude it from comparisons in the rest of this section.

6.2 Weighted aggregates

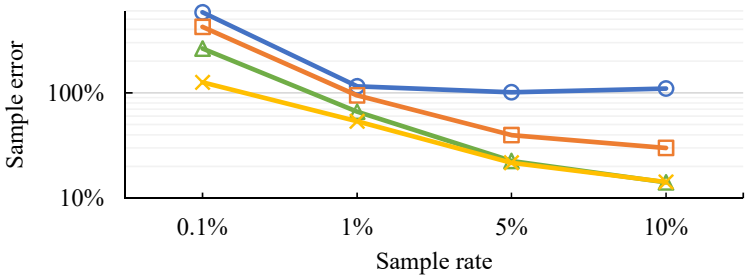
When multiple aggregates are involved, they are not necessarily equally important to the user. CVOPT allows the user to assign a greater weight to one aggregate over the others, leading to an improved quality for the answer, based on the user's need. We conducted experiments with query AQ2 and query B1. Each query has two aggregations, Agg1 and Agg2, with the weights denoted by w_1 and w_2, respectively. We use CVOPT to draw 3 samples with different weighting profiles: $(w_1, w_2) = \{(0.1, 0.9), (0.5, 0.5), (0.9, 0.1)\}$, as the user favors Agg2 over Agg1 in the first case and favors Agg1 in the third case. The second case is equal to default, non-weighted setting. Results are presented in figure 2. From the left to the right side, as w_1 increases and w_2 decreases, the average error of Agg1 decreases and of Agg2 increases. The results in both queries show CVOPT's ability to create a sample that better fits the user's priority. While previous heuristic works cannot systematically support weighted aggregates, we find this feature is practically useful in calibrating the sample.

6.3 Sample's usability

On Sample Rate: It is well known that a higher sample rate generally improves the sample's accuracy in query processing. Nevertheless, we compare the accuracy of the different methods for queries AQ2 and B2 under different sample rates. Figure 3 shows that CVOPT outperforms its peers



(a) `MASG` query AQ2 answered by samples with various sample rates.



(b) `SASG` query B2 answered by samples with various sample rates.

Uniform CS RL CVOPT

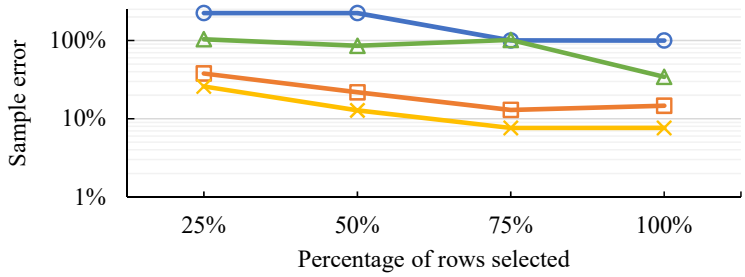
Figure 3: Sensitivity of maximum error to sample size.

	AQ3	AQ3.a	AQ3.b	AQ3.c	AQ5	AQ6
`Uniform`	98.4	21.0	21.4	18.0	99.6	100.0
`CS`	2.5	5.8	2.9	2.8	3.9	0.9
`RL`	5.4	9.5	6.9	5.6	4.3	3.5
`CVOPT`	1.5	4.4	2.4	1.9	2.3	0.8

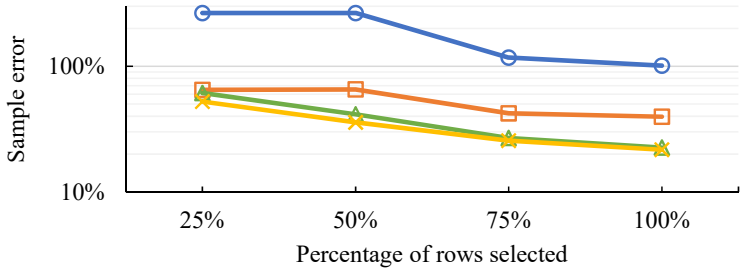
Table 5: Average error of multiple queries answered by one materialized sample, showing the reusability of the sample.

at nearly all sample rates of the study.

On Predicate Selectivity and Group-by Attributes: Queries commonly come with selection predicates, *i.e.,* the `WHERE` clause in a SQL statement. Since samples are pre-constructed, the same sample is used no matter what the selection predicate is. We study the sample's usability on the selectivity of the predicate. Query AQ3 has a predicate: `WHERE HOUR(local_time) BETWEEN <low> AND <high>`. By changing the filter conditions, we generated 4 queries AQ3.a, AQ3.b, AQ3.c and AQ3, with selectivity of 25%, 50%, 75% and 100%, respectively. All queries are answered by the materialized sample optimized for query AQ3. Similarly, we have queries B2.a, B2.b, B2.c, and B2, with controllable predicate parameters to table `Bikes`. Figure 4 shows the results. Although the samples are able to serve different queries, we observe the effect of selectivity upon the accuracy. Overall, the greater the selectivity, the lesser is the error due to sampling. For each predicate query, `CVOPT` has a lower error than `CS` and `RL`. Table 5 shows the average error of a group of six different queries using the materialized sample optimized for AQ3. Note that AQ5 and AQ6 use



(a) SASG queries with different predicates on OpenAQ.



(b) SASG queries with different predicates on Bikes.

Figure 4: Maximum error of multiple queries with different predicates answered by one materialized sample, showing the reusability of the sample

different **WHERE** clauses that are also different from those used by AQ3 and AQ3.*. AQ6 also uses a different set of **GROUP-BY** attributes. For all six queries, CVOPT performs well with good accuracy and is consistently better than other methods.

6.4 Multiple Group-by Query

WITH CUBE is an extension of group-by clause in SQL that generates multiple grouping sets of given attributes in a single query. For example, GROUP BY A, B WITH CUBE will generate grouping sets of (A, B), (A), (B), and (). WITH CUBE is powerful in helping user to easily and efficiently compute aggregations with a large number of combinations of group-by. We conduct experiments with SAMG queries AQ7 and B3 and MAMG queries AQ8 and B4. The queries have the grouping sets of two attributes for multiple group-by. OpenAQ has 38 countries and 7 parameters, so cube with these two attributes will generate upto 312 groups. Bikes has 619 stations and 3 years of collection, and therefore cube with *from_station_id* and *year* leads upto 2480 groups.

All methods RL, CS, and CVOPT, can sample in the presence of multiple groupings; for CS this is the scaled congressional sampling method and for RL it is the hierarchical partitioning method. Both CS and RL adopt a heuristic approach, which we implemented as described in their algorithm. Accuracies of different samplers are shown in Figure 5. We note that CVOPT performs significantly better than Uniform and RL and is consistently better than CS.

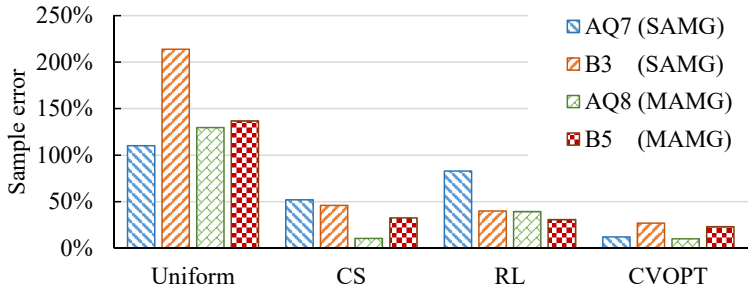


Figure 5: Maximum error of `CUBE` group-by queries.

	`OpenAQ` (40 GB)		`OpenAQ-25x` (1 TB)	
	Precomputed	Query	Precomputed	Query
`Full Data`	-	2881.68	-	60565.49
`Uniform`	913.95	40.31	9060.53	294.40
`Sample+Seek`	2309.99	44.92	58119.91	821.46
`CS`	3854.61	55.87	87036.11	1091.60
`RL`	4311.36	54.43	99365.67	952.37
`CVOPT`	4263.07	59.99	98081.49	1095.18

Table 6: The sum of the CPU time (in seconds) at all nodes for sample precomputing and query processing with 1% sample for query AQ1 over `OpenAQ` and `OpenAQ-25x`.

6.5 CPU Time

We measure the CPU time cost for sample precomputation and query processing, as the sum of the CPU cost at all nodes in a Hadoop cluster we used. The cluster includes a master node and 3 slave nodes. Each node has 2.70 GHz x4 CPU with 10 GB memory. For timing purposes, we generate a 1 TB dataset, `OpenAQ-25x`, by duplicating `OpenAQ` 25 times.

Table 6 shows the CPU time for precomputing the 1% samples and query processing to answering query AQ1. Due to the two scans required, stratified random samples, `Sample+Seek`, `CS`, `RL` and `CVOPT`, are more expensive than `Uniform`, which requires only a single scan through data.

Query processing using samples could reduce CPU usage by 50 to 300 times. `CS` and `Sample+Seek` have slightly less CPU time. `RL` has comparable CPU time cost. Since `Uniform` missed a number of groups, its query time is smaller. `CVOPT` consumed only about 2% of the CPU compared to query over the base table, reducing the CPU time from 0.8 hour to less than a minute for the 40GB `OpenAQ`, and from 17 hours to 18 minutes for the 1TB `OpenAQ-25x`.

For both data sets, the precomputation time of `CVOPT` is about 1.5x the cost of running the original query on the full data, indicating that with only *two* queries, the precomputation time can be amortized to be cheaper than running the original queries on the full data.

6.6 Experiments with `CVOPT-INF`

Our experiments show that `CVOPT`, which optimizes for the ℓ_2 norm of the CVs, leads to smaller errors at higher percentiles than `CS` and `RL`. We now consider `CVOPT-INF`, which minimizes for the maximum of CVs of all estimates (ℓ_∞ norm). Our results on the accuracy of `CVOPT-INF` on queries

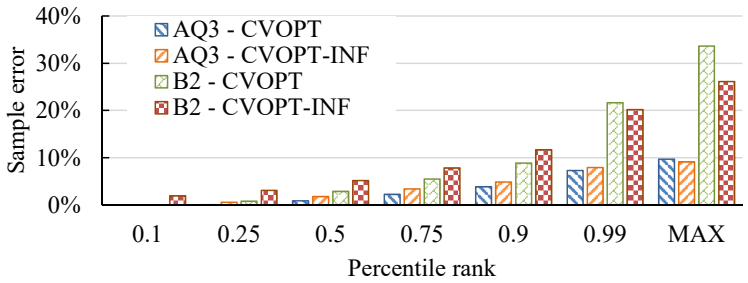


Figure 6: Comparison accuracy from `CVOPT` and `CVOPT-INF` for `SASG` queries AQ3 and B2.

AQ3 and B2 are shown in Figure 6. Consistent with theoretical predictions, `CVOPT-INF` has a lower maximum error than `CVOPT` on both queries. At the same time, `CVOPT-INF` has a worse error than `CVOPT` at the 90^{th} percentile and below. Overall, this shows that `CVOPT-INF` can be considered when the user is particularly sensitive to the maximum error across all groups. Otherwise, `CVOPT` with ℓ_2 optimization provides robust estimate for a large fraction of the groups, with a small error across a wide range of percentiles.

7 Other Related Work

Random sampling has been widely used in approximate query processing, for both static [6, 7, 21, 3, 8, 14, 23] and streaming data [12, 11, 27, 2]. Uniform sampling is simple and can be implemented efficiently (e.g. using reservoir sampling [25]), but does not produce good estimators for groups of low volume or large variance. Ganti et al. [10] address low selectivity queries using workload-based sampling, such that a group with a low selectivity is sampled as long as the workload includes queries involving that group. Different techniques have been used in combination with sampling, such as indexing [9, 26, 4] or aggregate precomputation [19].

Chaudhuri et al. [5] formulate approximate query processing as an optimization problem. Their goal is to minimize the ℓ_2 norm of the relative errors of all queries in a given workload. Their approach to group-by queries is to treat every group derived from every group-by query in the workload as a separate query. In doing so, their technique does not handle overlap between samples suited to different groups. In contrast, our framework considers the overlaps and interconnections between different group-by queries in its optimization.

Kandula et al. [13] consider queries that require multiple passes through the data, and use random sampling in the first pass to speed up subsequent query processing. This work can be viewed as query-time sampling while ours considers sampling before the full query is seen. Further, [13] does not provide error guarantees for group-by queries, like we consider here. A recent work [17] has considered stratified sampling on streaming and stored data, addressing the case of full-table queries using an optimization framework. This work does not apply to group-by queries.

8 Conclusion

We presented `CVOPT`, a framework for sampling from a database to accurately answer group-by queries with a provable guarantee on the quality of allocation, measured in terms of the ℓ_2 (or ℓ_∞) norm of the coefficient of variation of the different per-group estimates. Our framework is quite

general, and can be applied anytime it is needed to optimize jointly for multiple estimates. Choosing the ℓ_2 of the CVs, larger errors are penalized heavily, which leads to an allocation where errors of different groups are concentrated around the mean. Choosing the ℓ_∞ of the CVs leads to a lower value of the maximum error than the ℓ_2, at the cost of a slightly larger mean and median error. There are many avenues for further research, including (1) incorporating joins into the sampling framework (2) exploring ℓ_p norms for values of p other than $2, \infty$, (3) handling streaming data.

References

[1] S. Acharya, P. Gibbons, and V. Poosala. Congressional samples for approximate answering of group-by queries. In *SIGMOD*, pages 487–498, 2000.

[2] N. K. Ahmed, N. Duffield, T. L. Willke, and R. A. Rossi. On sampling from massive graph streams. *Proc. VLDB Endow.*, 10(11):1430–1441, Aug. 2017.

[3] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *Proc. SIGMOD*, pages 539–550, 2003.

[4] S. Chaudhuri, G. Das, M. Datar, R. Motwani, and V. R. Narasayya. Overcoming limitations of sampling for aggregation queries. In *ICDE*, pages 534–542, 2001.

[5] S. Chaudhuri, G. Das, and V. Narasayya. Optimized stratified sampling for approximate query processing. *ACM Trans. Database Syst.*, 32(2), 2007.

[6] S. Chaudhuri, B. Ding, and S. Kandula. Approximate query processing: No silver bullet. In *SIGMOD*, pages 511–519, New York, NY, USA, 2017. ACM.

[7] Y. Chen and K. Yi. Two-level sampling for join size estimation. In *SIGMOD*, pages 759–774, 2017.

[8] W. G. Cochran. *Sampling Techniques*. John Wiley & Sons, New York, third edition, 1977.

[9] B. Ding, S. Huang, S. Chaudhuri, K. Chakrabarti, and C. Wang. Sample + seek: Approximating aggregates with distribution precision guarantee. In *SIGMOD*, pages 679–694, 2016.

[10] V. Ganti, M. Lee, and R. Ramakrishnan. ICICLES: self-tuning samples for approximate query answering. In *VLDB*, pages 176–187, 2000.

[11] P. J. Haas. Data-stream sampling: Basic techniques and results. In *Data Stream Management*, pages 13–44. Springer, 2016.

[12] B. Hentschel, P. J. Haas, and Y. Tian. Temporally-biased sampling for online model management. In *EDBT*, pages 109–120, 2018.

[13] S. Kandula, A. Shanbhag, A. Vitorovic, M. Olma, R. Grandl, S. Chaudhuri, and B. Ding. Quickr: Lazily approximating complex adhoc queries in bigdata clusters. In *SIGMOD*, pages 631–646, 2016.

[14] S. L. Lohr. *Sampling: Design and Analysis*. Duxbury Press, 2nd edition, 2009.

[15] R. Nambiar and M. Poess. The making of tpc-ds. In *PVLDB*, pages 1049–1058, 2006.

[16] J. Neyman. On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society*, 97(4):558–625, 1934.

[17] T. D. Nguyen, M. Shih, D. Srivastava, S. Tirthapura, and B. Xu. Stratified random sampling over streaming and stored data. In *EDBT*, pages 25–36, 2019.

[18] http://openaq.org.

[19] J. Peng, D. Zhang, J. Wang, and J. Pei. Aqp++: Connecting approximate query processing with aggregate precomputation for interactive analytics. In *SIGMOD*, pages 1477–1492, 2018.

[20] P. Rösch and W. Lehner. Sample synopses for approximate answering of group-by queries. In *EDBT*, pages 403–414, 2009.

[21] S. K. Thompson. *Sampling*. Wiley, 3rd edition, 2012.

[22] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: A warehousing solution over a map-reduce framework. *Proc. VLDB Endow.*, 2(2):1626–1629, Aug. 2009.

[23] Y. Tillé. *Sampling Algorithms*. Springer-Verlag, 1st edition, 2006.

[24] https://www.divvybikes.com/system-data.

[25] J. S. Vitter. Optimum algorithms for two random sampling problems. In *Proc. FOCS*, pages 65–75, 1983.

[26] L. Wang, R. Christensen, F. Li, and K. Yi. Spatial online sampling and aggregation. *Proc. VLDB Endow.*, 9(3):84–95, Nov. 2015.

[27] X. Zhang, J. Wang, and J. Yin. Sapprox: Enabling efficient and accurate approximations on sub-datasets with distribution-aware online sampling. *Proc. VLDB Endow.*, 10(3):109–120, Nov. 2016.

Query AQ 1 Changing of *bc* overtime for each country.

```
WITH bc18 AS (
  SELECT country, AVG(value) AS avg_value,
         COUNT_IF(value > 0.04) AS high_cnt
  FROM   openaq WHERE  parameter = 'bc'
    AND  YEAR(local_time) = 2018
  GROUP BY country ),
bc17 AS (
  SELECT country, AVG(value) AS avg_value,
         COUNT_IF(value > 0.04) AS high_cnt
  FROM   openaq WHERE  parameter = 'bc'
    AND  YEAR(local_time) = 2017
  GROUP BY country )
SELECT country,
  bc18.avg_value - bc17.avg_value AS avg_incre,
  bc18.high_cnt - bc17.high_cnt AS cnt_incre
FROM bc18 JOIN bc17
ON bc18.country = bc17.country
```

Query AQ 2 MASG query to `OpenAQ` table.

```
SELECT country, parameter, unit,
       SUM(value) agg1, COUNT(*) agg2
FROM OpenAQ
GROUP BY country, parameter, unit
```

Query B 1 MASG query to `Bikes` table.

```
SELECT from_station_id,
       AVG(age) agg1, AVG(trip_duration) agg2
FROM Bikes      WHERE age > 0
GROUP BY from_station_id
```

Query AQ 3 Average value.

```
SELECT country, parameter, unit, AVG(value)
FROM OpenAQ
WHERE HOUR(local_time) BETWEEN 0 AND 24
GROUP BY country, parameter, unit
```

Query B 2 Average trip duration from each station.

```
SELECT from_station_id, AVG(trip_duration)
FROM Bikes      WHERE trip_duration > 0
GROUP BY from_station_id
```

Query AQ 4 Average carbon monoxide.

```
SELECT AVG(value),
       country,
       CONCAT(month, '_', year)
FROM (SELECT value,
             MONTH(local_time) AS month,
             YEAR(local_time) AS year,
             country
      FROM  OpenAQ      WHERE parameter = 'co' )
GROUP BY country, month, year
```

Query AQ 5 Average measurement for each parameter of the countries in northern hemisphere.

```
SELECT  country, parameter, unit,
        AVG(value) average
FROM OpenAQ     WHERE latitude > 0
GROUP BY country, parameter, unit
```

Query AQ 6 Count the number of times the measurement of each parameter is higher than 0.5 in Vietnam.

```
SELECT
    parameter, unit,
COUNT(IF(value > 0.5, 1, 0)) count
FROM {input_table}      WHERE country = "VN"
GROUP BY parameter, unit
```

Query AQ 7 Single aggregate, multiple group-by, OpenAQ.

```
SELECT country, parameter, SUM(value)
FROM OpenAQ
GROUP BY country, parameter WITH CUBE
```

Query B 3 Single aggregate, multiple group-by, Bikes

```
SELECT  from_station_id, year,
        SUM(trip_duration)
FROM Bikes      WHERE age > 0
GROUP BY from_station_id, year WITH CUBE
```

Query AQ 8 Multiple aggregate, multiple group-by, OpenAQ.

```
SELECT country, parameter,
       SUM(value), SUM(latitude)
FROM OpenAQ
GROUP BY country, parameter WITH CUBE
```

Query B 4 Multiple aggregate, multiple group-by, `Bikes`

```
SELECT  from_station_id, year,
        SUM(trip_duration), SUM(age)
FROM Bikes
GROUP BY from_station_id, year WITH CUBE
```